WWW.SCIAM-MAGAZINE.NAROD.RU

Анализ результатов первого турнира по компьютерной игре «Бой в памяти»

А. К. ДЬЮДНИ

«БОЙ В ПАМЯТИ» — игра, участники которой, компьютерные программы, пытаются уничтожить друг друга, привлекла к себе внимание в конце прошлого года, когда в музее вычислительной техники в Бостоне (шт. Массачусетс) состоялся первый международный турнир, организованный ее энтузиастами. В турнире принимала участие 31 программа, три из них, оказавшиеся самыми боеспособными, стали призерами соревнований. Абсолютным чемпионом была объявлена программа MICE (мышка). Ее автору Ч. Уэнделлу из Рочестера был вручен приз — специально изготовленная декоративная панель с элементами памяти для игры «Бой в памяти» в ее первоначальной версии, относящейся к тому времени, когда появились первые игровые программы, способные вести бой и выполнявшиеся на компьютере CDC 6600.

Игра «Бой в памяти» уже дважды была темой статей в рубрике «Занимательный компьютер» (см. «В мире науки», 1984, № 7 и 1985, № 5). Созданные людьми боевые программы начинают самостоятельную жизнь, вступая в сражение с противником на поле компьютерной памяти. Область памяти, отведенная под поле боя, называется CORE (сердечник) — так называлась оперативная память компьютера в те времена, когда ее ячейки представляли собой миниатюрные колечки из ферромагнитных сплавов. Эта игра стала настолько популярной, что ее любители организовали свое международное общество. Недавно это общество несколько изменило правила игры и теперь игроки будут придерживаться ее новой версии.

Основу игры составляют боевые программы, написанные на специальном языке низкого уровня под названием Редкод (Redcode). Именно такие программы и выступали на недавнем турнире. При помощи 10 простых команд программа может перемещать информацию из одной ячейки памяти в другую, складывать и вычитать числа, изменять порядок выполнения команд и даже выполнять несколько команд одновременно (см. рисунок на с. 98). Рассмотрим, например, одну из основных команд на перемещение информации MOV. Она состоит из трех частей — кода команды и двух адресов — и занимает одну ячейку памяти. В общем виде команда записывается как MOV A B. Если A, скажем, равно 102, а B равно -5, то программа обратится к ячейке, отстоящей от данной ячейки на 102 адреса, и, прочтя информацию, которая там хранится, поместит ее в ячейку, расположенную на 5 адресов до ячейки, хранящей команду MOV.

Простейшая программа на языке Редкод состоит всего из одной команды MOV 0 1. Эта программа JMP (будем называть ее чертенком) перемещает содержимое ячейки с относительным адресом 0 (т. е. саму команду MOV) в ячейку с относительным адресом 1, т. е. в ячейку, расположенную на один адрес впереди. Команды языка Редкод обычно выполняются последовательно. Это означает, что после выполнения команды MOV 0 1 компьютер попытается выполнить команду, записанную в следующей ячейке. Но там теперь находится команда MOV 0 1, только что скопированная туда из ячейки с предшествующим адресом. В результате чертенок прыгает от адреса к адресу по памяти компьютера, сокрушая все на своем пути. Позади он оставляет след из команд MOV 0 1.

Чертенок может похитить самое главное в программе противника, а именно право на выполнение. Чтобы понять, как это происходит, предположим, что у нас есть боевая программа, выполняющаяся, как обычно, в том порядке, в каком расположены ее команды. Чертенок вторгается в пределы этой программы, заменяя ее команды бесконечной последовательностью MOV 0 1. Рано или поздно наша программа, наверное, передаст управление на поврежденный участок. Но в этом случае она сразу превратится в нового чертенка. Выступая под старым флагом, она вынуждена теперь следовать на поводу у противника, чертенка, до тех пор, пока сражение не кончится.

Чтобы обезопасить себя от нападения чертенка, программа для «Боя в памяти» должна по крайней мере содержать в себе специальный барьер. Этот предохранительный барьер состоит из двух команд, выполняющихся циклически:

```
MOV ≠ 0 − 1
JMP − 1
```

Первая команда перемещает целое число 0, записанное как $\neq 0$, в ячейку с относительным адресом -1. Другими словами, при каждом выполнении команды MOV ячейка, расположенная непосредственно «над» ней (только оттуда следует ожидать вторжения чертенка), заполняется числом 0. Вторая команда JMP является командой перехода, или передачи управления. После того как она выполнена, управление переходит на ячейку с относительным адресом -1, т. е. на ячейку, расположенную непосредственно над командой MOV. При каждом цикле выполнения программа сбрасывает нулевую бомбу на то место, куда перед этим мог добраться чертенок, — на ячейку, прилегающую к барьеру сверху. В результате чертенок уничтожается.

Игра «Бой в памяти» основана на двух правилах. Первое заключается в том, что программы-соперники выполняют свои команды по очереди. За чередованием ходов следит управляющая система Марс (MARS — от слов Memory Array Redcode Simulator). Из этого несколько неуклюжего военного названия следует, что Марс имитирует действия компьютера. Марс постоянно изменяет содержимое ячеек оперативной памяти в соответствии с выполняемыми командами языка Редкод. Каждой стороне предоставляется право выполнения лишь одной команды, после чего выполняется команда программы противника. Второе правило состоит в том, что когда одна из программ не может выполнить свою очередную команду, то эта программа считается проигравшей.

Выполнение программы может разделиться на несколько линий. Если в программе, написанной на языке Редкод, встречается команда SPL A, то дальнейшее выполнение программы как бы идет по двум линиям. Одна линия начинается с команды, непосредственно следующей за SPL A, а другая перескакивает на команду с относительным адресом A. К сожалению, Марс не может выполнить обе эти команды одновременно, одну команду система выполняет на одном ходе, предоставленном данной программе, а вторую — на следующем

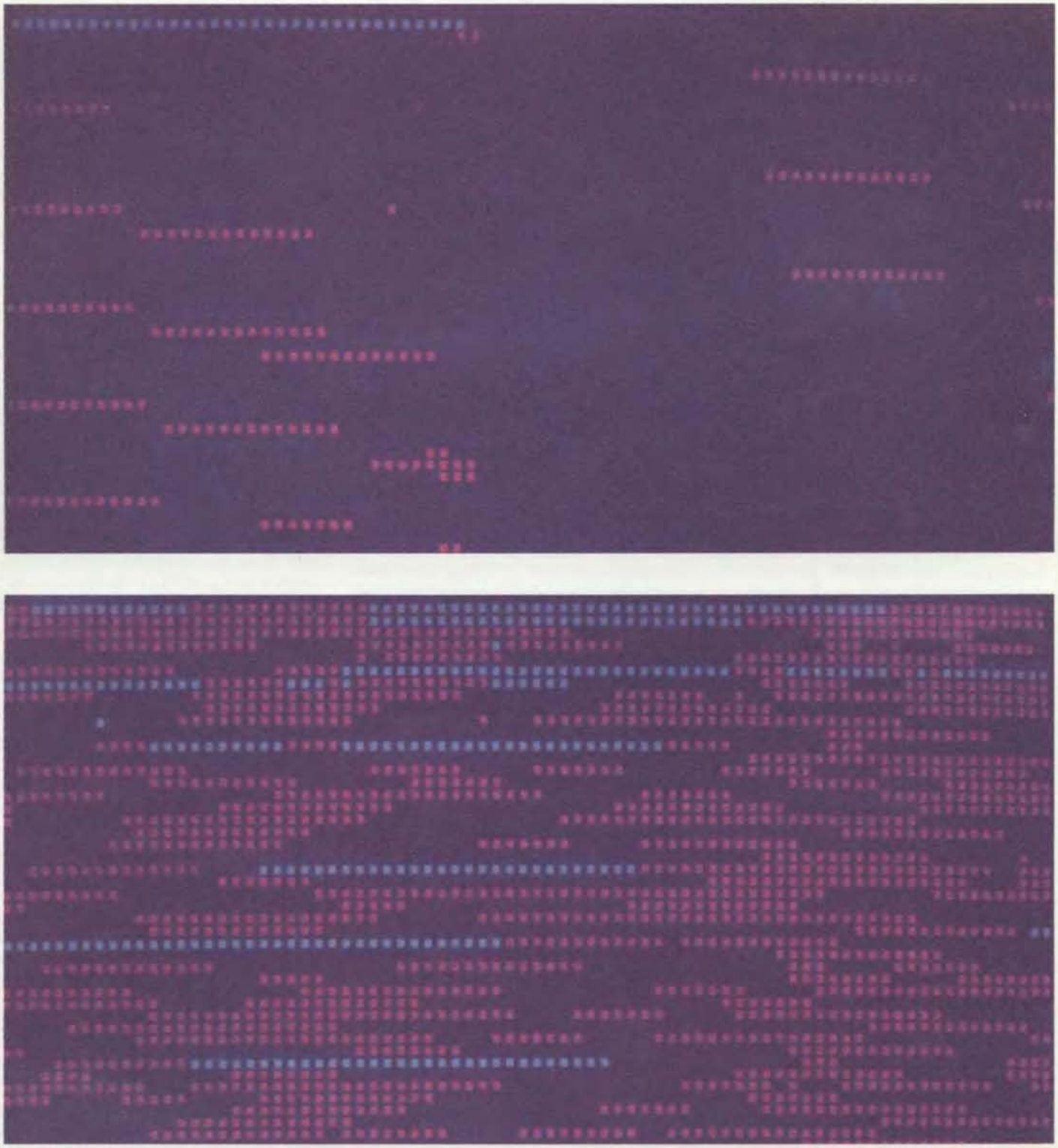
WWW.SCIAM-MAGAZINE.NAROD.RU

ходе. Таким образом сводится на нет, казалось бы, важное преимущество: чем больше у программы различных линий выполнения, тем медленнее выполняется каждая из этих линий. Но в итоге подобное замедление оказывается оправданным. Если у программы есть несколько линий выполнения, то она считается проигравшей лишь в том случае, когда ни одна из этих линий не может быть продолжена. Эта ситуация возникает, когда система Марс, ожидая найти выполнимую команду языка Редкод, находит лишь то, что можно, образно выражаясь, назвать воронками от взорванных снарядов и бомб.

Чтобы проиллюстрировать возможности команды SPL, рассмотрим, к примеру, пять первых команд моей собственной программы, принявшей участие в турнире «Бой в памяти». Я назвал ее COMMANDO (десантник-диверсант) по причинам, о которых скажу несколько позже.

```
MOV  ≠ 0 – 1
JMP  – 1
SPL  – 2
MOV 10 113
SPL 112
  .
  .
  .
```

В первых двух командах читатели, наверное, узнали барьер, предохра-



Одна из начальных и одна из завершающих стадий в сражении между программами MICE (красный) *и CHANG1* (синий)

WWW.SCIAM-MAGAZINE.NAROD.RU

няющий программу от вторжения чертенка. Выполнение самой программы начинается с третьей команды: SPL — 2. На следующих двух ходах, предоставленных программе COMMANDO, будут выполнены первая и четвертая команды, а затем при следующих двух очередных ходах — вторая и пятая команды. Каждая линия программы развивается независимо от другой и с вдвое меньшей скоростью, если можно так выразиться. В рассматриваемом нами примере COMMANDO сначала запускает свой оборонительный фрагмент, который в дальнейшем работает самостоятельно. Затем она запускает своего собственного чертенка (начиная с ячейки, отстоящей на 10 адресов от второй команды MOV) в далекую ячейку (отстоящую на 113 адресов). Второй чертенок запускается следующей командой SPL.

Остальные команды программы COMMANDO предназначены для того, чтобы скопировать всю программу в другую область, которая удалена на 100 адресов от ее теперешнего положения. Новая копия программы, как десантник, только что приземлившийся на территории противника, запускается командой JMP в исходной программе. Старый экземпляр COMMANDO, за исключением барьерного фрагмента, прекращает свое существование и больше не выполняется. Затем описанный процесс повторяется для нового экземпляра программы, который в свою очередь копируется на новое место в памяти.

Ну а насколько успешно выступила программа COMMANDO против своих соперников? Турнир был организован так, чтобы между программами-участниками (а их, как уже отмечалось, было 31) состоялось как можно больше поединков. Розыгрыш по полной круговой системе, когда каждый участник встречается со всеми другими участниками, потребовал бы проведения 465 поединков и занял бы слишком много времени. Поэтому участники были поделены произвольно на две приблизительно равные группы: группа I и группа II. Затем внутри каждой группы был проведен турнир по полной круговой системе.

Известие о том, что COMMANDO стала победителем турнира в группе II, я встретил со смешанным чувством. С одной стороны, я был очень горд тем, что мое кибернетическое творение так хорошо проявило себя в действии. В то же время я испытывал некоторое беспокойство при мысли о том, что моя программа может оказаться победителем соревнований. Дело в том, что я согласился выступить в роли комментатора финальных состязаний и чувствовал, что мне трудно будет сохранить объективность.

Четыре лучшие программы из каждой группы выступили затем в новом круговом турнире. Победителями в этом турнире оказались три программы: CHANG1 — автор М. Чанг из Флорал-Парка (шт. Нью-Йорк) и две программы Ч. Уэнделла MIDGET и MICE. Моя программа COMMANDO, получив смертельное ранение, пала смертью храбрых. Интересно сложился финал, выигранный программой MICE. MIDGET и MICE свели свои поединки с CHANG1 вничью, но MICE получила решающее выигрышное очко, добившись победы в поединке с MIDGET.

Матч между каждой парой финалистов состоял из четырех боев, проводившихся один за другим. Поединки были ограничены по времени: каждая сторона могла сделать не более 15 тыс. ходов (команд), что приблизительно соответствует двум минутам машинного времени. Две встречающиеся в поединке боевые программы помещались в выбранные случайным образом непересекающиеся области памяти, после чего они запускались. Получилось так, что бои, проведенные между всеми парами программ, закончились с одним и тем же результатом. В частности, все бои между программами MICE и CHANG1 завершились вничью.

Очень интересно наблюдать за развитием боя. На турнире использовался дисплей, на котором отображалось поле боя в виде последовательности полос с клеточной структурой (см. рисунок на с. 97). Каждая клетка представляла одну ячейку (и один адрес) в памяти, причем последняя клетка в нижнем ряду была соседней по отношению к первой клетке в верхнем ряду. Таким образом, поле боя было замкнутым. Программа, которой предоставлялся первый ход, в исходном положении занимала область памяти, начинавшуюся с адреса 0, и ее команды последовательно заполняли прилегающие ячейки. Клетки, занятые этой программой, окрашивались в голубой цвет. Программа-соперник занимала произвольно выбранную область ячеек, не пересекавшуюся с областью, занятой первой программой. На экране эта область окрашивалась в ярко-красный цвет. А вообще цвет любой клетки на дисплее определялся тем, какая из соперничавших программ изменила ее содержимое последней. Таким образом, перед глазами наблюдателя разворачивалась картина сражения.

На темно-синем фоне экрана MICE и CHANG1 переползали с места на место, запускали чертиков, разбрасывали бомбы и воспроизводили сами себя (самопроизвольным делением).

КОМАНДА	ОБОЗНАЧЕНИЕ	ОПЕРАНДЫ		ПОЯСНЕНИЯ
Определить	DAT		B	Невыполняемая команда; B — значение элемента данных
Переместить	MOV	A	B	Переместить содержимое ячейки A в ячейку B
Сложить	ADD	A	B	Сложить содержимое ячеек A и B
Вычесть	SUB	A	B	Вычесть содержимое ячейки A из содержимого ячейки B
Перейти	JMP	A		Передать управление в ячейку A
Перейти, если 0	JMZ	A	B	Передать управление в ячейку A, если в ячейке B находится 0
Перейти, если больше 0	JMN	A	B	Передать управление в ячейку A, если содержимое ячейки B больше 0
Уменьшить и перейти, если больше 0	DJN	A	B	Вычесть 1 из содержимого ячейки B и передать управление по адресу A, если содержимое B становится больше 0
Сравнить	CMP	A	B	Сравнить содержимое ячеек A и B; если они равны, пропустить следующую команду
Разветвить	SPL	A		Разветвить выполнение команд: выполнить следующую команду и команду в ячейке A

Сведения о Редкоде, языке ассемблера для игры «Бой в памяти»

WWW.SCIAM-MAGAZINE.NAROD.RU

Проследим за тем, как развивались события в одном типичном бою между этими программами. CHANG1 возникла в виде голубой полоски в левом верхнем углу экрана, а появление на свет программы MICE было ознаменовано красной полоской, возникшей несколько ближе чем на полпути к нижнему краю экрана. Почти сразу же и очень быстро «мышки» начали разбегаться по экрану.

Одна из самых коротких известных мне самовоспроизводящихся программ MICE состоит всего из восьми команд, две из которых создают новую копию программы в области памяти, отстоящей на 833 адреса от положения, занимаемого ею на момент копирования (см. рисунок справа). Эти две команды демонстрируют некоторые дополнительные возможности языка Редкод:

loop MOV @ *ptr* < 5
DJN *loop ptr*

Здесь слово *loop* (цикл) представляет собой просто метку, обозначающую адрес (в данном случае ячейки, в которой содержится команда MOV). Благодаря использованию меток написание программы на Редкоде значительно упрощается. По команде DJN (уменьшение и переход к ненулевому значению операнда) происходит передача управления на команду, помеченную меткой *loop*, если значение величины, хранящейся в ячейке с другим адресом (помеченным меткой *ptr*), не равно нулю. Значок @ означает способ адресации, называемый косвенным: при выполнении команды MOV перемещается не содержимое ячейки, помеченной как ptr, а содержимое содержимого, если можно так выразиться. Число, хранящееся по адресу ptr, является адресом той ячейки, содержимое которой следует переместить. В данном случае это содержимое представляет собой одну из команд программы MICE.

Число, хранящееся в ячейке с адресом ptr, все время изменяется благодаря функции уменьшения, содержащейся в команде DJN. Исходное значение этого числа равно последнему адресу программы, затем оно постепенно уменьшается до 0 и здесь цикл копирования завершается. Аналогичным образом адреса, по которым должны разместиться команды программы на новом месте, также заданы косвенно. Ячейка с относительным адресом 5 первоначально хранит число 833, и первая команда, перемещаемая программой, приземлится в ячейке, отстоящей на 832 адреса от команды MOV. Как показывает значок <, адрес назначения уменьшается при каждом выполнении команды MOV. Мышки копируют себя как бы от хвоста к голове.

```
                CHANG1
        MOV     #0      –1
        JMP     –1
        DAT             +9
start   SPL     –2
        SPL     4
        ADD     #–16    –3
        MOV     #0      @–4
        JMP     –4
        SPL     2
        JMP     –1
        MOV     0       1
```

```
                MICE
ptr     DAT     #0
start   MOV     #12     ptr
loop    MOV     @ptr    <5
        DJN     loop    ptr
        SPL     @3
        ADD     #653    2
        JMZ     –5      –6
        DAT     833
```

Участники чемпионата игр «Бой в памяти»

Команда SPL (от англ. split — разветвление), следующая сразу же за циклом копирования, передает управление только что созданному экземпляру программы MICE. Но вслед за рождением отпрыска старый экземпляр программы выполняется сначала. В принципе количество отпрысков, порождаемых подобной программой, не ограничено. И каждая новая программа работает, следуя все той же процедуре. Действительно, получаются такие же мышки!

Так было и в типичном сражении с программой CHANG1. Мышки размножались с невероятной быстротой. Вскоре весь экран был усеян маленькими красными полосками. Тем временем CHANG1 запустила своеобразную фабрику, производящую на свет чертиков и спускающую их «вниз по течению». Эта фабрика была построена всего лишь из трех команд:

SPL 2
JMP – 1
MOV 0 1

Когда управление достигает команды SPL, процесс выполнения команд разветвляется на две ветви. Одна из них начинается с команды MOV 0 1. Другая ветвь выполняет команду JMP — 1, которая начинает весь описанный процесс заново. Тем временем один чертенок уже покинул конвейер и отправился на охоту за мышками. Основная проблема, возникающая при массовом производстве чертиков, заключается в том, что большое число независимых линий выполнения замедляет каждый из выполняемых процессов. 1000 чертиков движутся в 1000 раз медленнее, чем один чертенок. Во всяком случае, роковая орда чертиков стала медленно спускаться из верхней части экрана в виде все удлинявшейся сплошной голубой полосы. Смогут ли они перехитрить мышек?

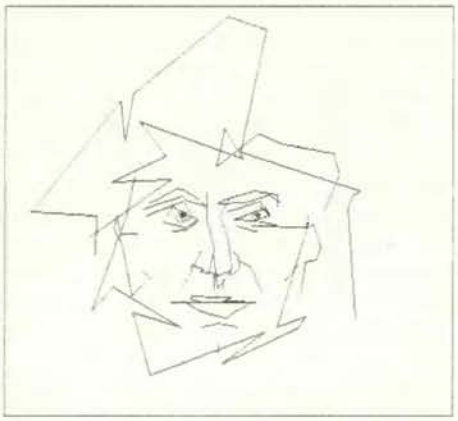
Пока размножались чертики, несколько экземпляров программы MICE, или несколько мышек, были убиты бомбами, брошенными программой CHANG1. Такая бомба представляет собой обычно число 0, «сбрасываемое» командой MOV предположительно на территорию противника. Ключевая команда в программе Чанга — это MOV ≠ ≠ 0 @ –4. Нулевая бомба падает в ячейку, адрес которой содержится в ячейке четырьмя адресами «выше» команды MOV. Адрес назначения постоянно увеличивается на 16 единиц, тем самым обеспечивается равномерный обстрел «по площади».

В то время как некоторые мышки погибали от бомб, чертики тоже начали причинять вред противнику. Однако в каждом экземпляре программы MICE заложена возможность самоубийства. Программа постоянно проверяет свою первую команду, которая представляет собой команду описания данных и должна содержать в себе число 0. Если там уже не 0, то программа совершает переход на (невыполнимую) команду описания данных и таким образом предпочитает погибнуть, чем стать жертвой крошечного неприятеля.

Но если одни мышки погибали под бомбами, а другие, так сказать, кончали жизнь самоубийством, чтобы избежать плена, то как же им удалось все-таки выжить? Объяснение, конечно, заключается в их чрезвычайной плодовитости. Ведь в конце концов многие новые экземпляры приземлились на голову противника. На самом деле незадолго до истечения времени поединка один из экземпляров программы MICE приземлился прямо на территории основной программы CHANG1 и уничтожил ее. Однако к этому моменту CHANG1 создала достаточно много чертиков, чтобы протянуть время до финального звонка. Поединок завершился с ничейным исходом.

Искусство создания программ для

WWW.SCIAM-MAGAZINE.NARODRU



Карикатура на среднестатистическое лицо

игры «Бой в памяти» находится еще в своей начальной стадии. Конечно, здесь будут сделаны свои открытия и будет наблюдаться постепенный неуклонный прогресс. Какому-нибудь изобретательному программисту, возможно, удастся открыть абсолютно надежное средство против угрозы чертиков, другой сумеет найти простой способ самовосстановления поврежденной программы.

Боевые программы следующих поколений, наверное, будут длиннее теперешних призеров, но зато их «живучесть» будет на несколько порядков величины выше. Они будут способны заниматься разведкой, оставлять ложный след и наносить внезапные сокрушительные удары по противнику. Эти тенденции, возможно, проявятся уже на втором международном турнире по игре «Бой в памяти», который должен состояться осенью этого года в музее вычислительной техники. А пока читатели имеют возможность солидно подготовиться, вложив в программы всю свою изобретательность и коварство.

Своим успехом прошлогодний турнир был в значительной степени обязан М. и Б. Кларксонам, а также Г. Беллу, возглавляющему музей вычислительной техники, и Оливеру Стримпелу, куратору музея. В заключение нужно, наверное, сказать несколько слов о самом музее.

Музей вычислительной техники в Бостоне, пожалуй, единственный в мире музей, экспозиция которого целиком посвящена компьютерам. Размещенный в перестроенном (и теперь роскошном) здании бывшего складского помещения на набережной, он демонстрирует компьютеры — от чудовищ, собранных на вакуумных электронных лампах, до персональных компьютеров, полностью предназначенных для игр. Стены помещения украшены удивительной графикой: здесь представлена полная компьютерная система NORAD SAGE и многие другие экспонаты как развлекательного, так и образовательного плана. Читатели, которые будут в Бостоне и посетят старинный корабль в бостонской бухте, могут заглянуть и в музей вычислительной техники, это буквально в двух шагах от бухты.

В ДЕКАБРЬСКОМ номере журнала за прошлый год в рубрике «Занимательный компьютер» была помещена статья с описанием программы, строящей карикатуры и созданной под влиянием работы С. Бреннан, сотрудницы фирмы Hewlett-Packard Laboratories в Пало-Альто (шт. Калифорния). В качестве входных данных в программу вводится цифровая версия портрета, карикатурное изображение которого должна построить программа. Введенный цифровой портрет программа сравнивает со среднестатистическим лицом, также хранящимся в памяти компьютера в цифровой форме. Затем программа увеличивает каждую черту представленного ей портрета с коэффициентом, пропорциональным тому расстоянию, на которое отстоят друг от друга соответствующие точки двух портретов. Если, например, ухо окажется несколько большим, чем у среднестатистического лица, то программа сделает его еще бóльшим, помножив расстояния между соответствующими точками на множитель искажения k.

Читателей, которые хотели бы сами реализовать программу FACEBENDER, возможно, несколько отпугивает перспектива перевода в цифровую форму своего собственного лица с фотографии. П. Макалузо из Уайт-Плейнза (шт. Нью-Йорк) в качестве исходного портрета для карикатуры использует среднестатистическое лицо. «Главное, — пишет Макалузо, — подобрать диапазон вариаций таким образом, чтобы он соответствовал размеру данной черты лица. Так, например, величина уха может варьировать в более широких пределах, чем ямочка на подбородке. Для каждой черты лицы мы просто строим «описывающий» ее прямоугольник, вычислив максимально возможные и минимально возможные координаты x и y для этой черты». В этих зафиксированных диапазонах фактор искажения выбирается с использованием датчика случайных чисел в ходе выполнения программы. Таким образом, пользуясь версией программы FACEBENDER, созданной Макалузо, можно без труда получить целую галерею портретов. Одна из построенных этой программой карикатур напоминает портрет Леонардо да Винчи. Она показана слева.

Читатель из Пасадина (шт. Калифорния), известный нам только по своим инициалам Д. М. И., сделал предложение, позволяющее избежать «безликости» — ужасного состояния программы, когда коэффициент искажения становится слишком большим. Все черты лица превращаются при этом в какое-то невообразимое и неузнаваемое птичье гнездо из многоугольников. Представим себе, что лицо, предназначенное для карикатуры, наложено на среднестатистическое лицо и что соответственные точки обоих портретов связаны между собой пружинками. Теперь, когда искажающая процедура пытается переместить представительные точки введенного лица, она встречает со стороны пружинок определенное сопротивление. На относительно небольшие искажения оно оказывает пренебрежимо малое воздействие, но при увеличении размеров искажения сопротивление пружинок растет и в конечном итоге не позволяет изображению стать «безликим».

Издательство **МИР** *предлагает:*

Э. Баннаи, Т. Ито
АЛГЕБРАИЧЕСКАЯ КОМБИНАТОРИКА: Ассоциативные схемы

Перевод с английского

Книга известных японских математиков, систематически излагающая новый подход к решению разнообразных задач алгебры и комбинаторики, основанный на применении ассоциативных схем. В последние годы интерес к этой тематике возрос в связи с приложениями в теоретической и прикладной математике.

Для математиков разных специальностей, аспирантов и студентов как учебное пособие по дискретной математике.

1987, 19 л. Цена 2 р. 50 к.

ИЗДАТЕЛЬСТВО МИР